Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Please amend the claims as follows:

1. (currently amended) A method for storing program code on a data processing device comprising:

splitting said program code into one or more blocks;

assigning each of said blocks a header containing a sequence number identifying which portion of said program code each of said blocks correspond to;

storing said one or more blocks of program code and said associated headers in locations of a non-volatile memory of the data processing device; and

generating a map of the locations where said blocks of program code are stored in the non-volatile memory of the data processing device during end-user operation of said data processing device.

2. (original) The method as in claim 1 further comprising:

combining said blocks of program code in an appropriate order in memory based on said sequence numbers and processing said program code in response to a specified event.

3. (original) The method as in claim 1 wherein said headers further comprise:

an error detection code to detect whether said program code in each of said blocks is valid.

Appl. No. 10/053,205
Amdt. dated May 2, 2006   2
Reply to the Office action of December 2, 2005

04676.P012

4. (previously presented) The method as in claim 3 wherein said error detection code is a cyclical redundancy checking (CRC) code.

5. (original) The method as in claim 1 further comprising:

upgrading said program code on said data processing device by replacing a specified number of said one or more blocks, said specified number being less that a total number of said blocks containing said program code.

6. (original) The method as in claim 1 further comprising:

verifying a signature associated with said program code prior to storing said program code.

7. (previously presented) The method as in claim 6 wherein verifying comprises:

applying a public key to said program code to generate the signature; and

comparing said signature to a signature generated via a private key.

8. (previously presented) The method as in claim 1 further comprising:

loading one of said blocks from non-volatile memory into volatile memory;

modifying a portion of said program code stored in said loaded block; and

storing said modified loaded block back to said non-volatile memory.

9. (previously presented) The method as in claim 8 further comprising:

re-calculating an error detection code associated with said modified loaded block to ensure that said program code contained in said modified loaded block is valid.

10. (currently amended) A method implemented by a service for maintaining control over stored one or more applications on a data processing device:

transmitting said one or more applications from a server to said data processing device concurrently with block allocation data indicating blocks on said data processing device into which said one or more applications are to be stored; and

maintaining a list of all subsequent data transactions performed with said data processing device during end-user operation of said device, said list usable by said server to construct a map of all applications stored on said data processing device.

11. (original) The method as in claim 10 further comprising:

generating said map of all applications stored on said data processing device prior to a new transaction with said data processing device.

12. (original) The method as in claim 11 wherein said new transaction is an application patch.

13. (original) The method as in claim 11 wherein said new transaction is a new application download.

14. (original) The method as in claim 12 further comprising:

transmitting said patch to said data processing device along with an indication of where said patch should overwrite a portion of said application.

Appl. No. 10/053,205
Amdt. dated May 2, 2006   4
Reply to the Office action of December 2, 2005

04676.P012

15. (original) The method as in claim 14 wherein said indication is an offset in memory.

16. (original) The method as in claim 15 further comprising:

loading said one or more blocks of program code into volatile memory,

identifying said portion of said program code with said offset and

overwriting said portion of said program code with said program code patch.

17. (original) The method as in claim 16 further comprising:

executing an error checking algorithm to ensure said one or more blocks of program code containing said program code patch are valid.

18. (original) The method as in claim 17 further comprising:

storing said one or more blocks of program code containing said program code patch to non-volatile memory.

19. (currently amended) A method comprising:

maintaining a program code map on a server indicating how program code is allocated among a plurality of non-volatile memory blocks on a data processing device;

using said program code map to facilitate modifications to said program code on said data processing device;

maintaining a list of transactions between said server and said data processing device during end-user operation of said data processing device; and

running an algorithm to construct said map in realtime using said list of

transactions, said algorithm being an algorithm executed on said data processing

device to store program code within said plurality of non-volatile memory blocks

of the data processing device.

20. (original) The method as in claim 19 wherein using said program

code map comprises:

calculating a location of a program code patch within said program code;

and

transmitting said program code patch to said data processing device

along with said location.

21. (original) The method as in claim 20 wherein said location is an offset

in memory.

22. (canceled)

23. (previously presented) The method as in claim 19 wherein said non-

volatile memory blocks are flash memory blocks.

24. (previously presented) The method as in claim 19 wherein using said

program code map comprises:

identifying a specific block within said non-volatile memory to be replaced;

transmitting said specific block to said data processing device;

said data processing device combining said block in volatile memory with one or more other blocks containing program code for a specific application and verifying a signature of said application; and

saving said specific block to non-volatile memory once said signature is verified.

25. (previously presented) The method as in claim 24 further comprising:

executing a cyclical redundancy checking (CRC) to verify that said program code within said specific block is valid.

26. (currently amended) A system comprising:

a server to transmit program code to a data processing device and to continually monitor during end-user operation of said data processing device (1) which program code is stored on said data processing device and (2) specific areas in a memory space in which said program code is stored on said data processing device, and

the server to transfer additional program code to said data processing device along with storage location data indicating where in said memory said additional program code should be stored.

27. (original) The system as in claim 26 wherein said storage location data comprises an offset in said memory of said data processing device indicating where within said memory said additional program code should be transmitted.

28. (previously presented) The system as in claim 26 wherein said storage location data comprises one or more non-volatile memory blocks where said additional program code should be stored.

29. (original) The system as in claim 28 wherein said additional program code comprises a patch to a current program stored on said data processing device.

30. (original) The system as in claim 26 wherein said data processing device is a wireless device.

31. (previously presented) The system as in claim 28 wherein said server transfers a cyclical redundancy checking (CRC) value with each block of program code transmitted to said data processing device.

32. (original) The system as in claim 29 wherein said server transfers an updated CRC value with said patch, said CRC value identifying data within a particular block in which said patch is required as being valid data.

33. (original) The system as in claim 29 wherein said server transfers an updated application signature usable by said data processing device to authenticate an application upgraded by said patch.